RUBY RANGES

Ranges occur everywhere: January to December, 0 to 9, lines 50 through 67, and so on. Ruby supports ranges and allows us to use ranges in a variety of ways:

- Ranges as Sequences
- Ranges as Conditions
- Ranges as Intervals

Ranges as Sequences:

The first and perhaps most natural use of ranges is to express a sequence. Sequences have a start point, an end point, and a way to produce successive values in the sequence.

Ruby creates these sequences using the **".."** and **"..."** range operators. The two-dot form creates an inclusive range, while the three-dot form creates a range that excludes the specified high value.

```
(1..5)        #==> 1, 2, 3, 4, 5
(1...5)       #==> 1, 2, 3, 4
('a'..'d')    #==> 'a', 'b', 'c', 'd'
```

The sequence 1..100 is held as a Range *object* containing references to two *Fixnum* objects. If you need to, you can convert a range to a list using the *to_a* method. Try following example:

```
#!/usr/bin/ruby

$, =", "   # Array value separator
range1 = (1..10).to_a
range2 = ('bar'..'bat').to_a

puts "#{range1}"
puts "#{range2}"
```

This will produce following result:

```
1, 2, 3, 4, 5, 6, 7, 8, 9, 10
bar, bas, bat
```

Ranges implement methods that let you iterate over them and test their contents in a variety of ways:

```
#!/usr/bin/ruby

# Assume a range
digits = 0..9

puts digits.include?(5)
ret = digits.min
puts "Min value is #{ret}"

ret = digits.max
puts "Max value is #{ret}"

ret = digits.reject {|i| i < 5 }
puts "Rejected values are #{ret}"

digits.each do |digit|
   puts "In Loop #{digit}"
```

```
end
```

This will produce following result:

```
true
Min value is 0
Max value is 9
Rejected values are 5, 6, 7, 8, 9
In Loop 0
In Loop 1
In Loop 2
In Loop 3
In Loop 4
In Loop 5
In Loop 6
In Loop 7
In Loop 8
In Loop 9
```

Ranges as Conditions:

Ranges may also be used as conditional expressions. For example, the following code fragment prints sets of lines from standard input, where the first line in each set contains the word *start* and the last line the word *end.*:

```
while gets
   print if /start/../end/
end
```

Ranges can be used in case statements:

```
#!/usr/bin/ruby

score = 70

result = case score
   when 0..40: "Fail"
   when 41..60: "Pass"
   when 61..70: "Pass with Merit"
   when 71..100: "Pass with Distinction"
   else "Invalid Score"
end

puts result
```

This will produce following result:

```
Pass with Merit
```

Ranges as Intervals:

A final use of the versatile range is as an interval test: seeing if some value falls within the interval represented by the range. This is done using ===, the case equality operator.

```
#!/usr/bin/ruby

if ((1..10) === 5)
  puts "5 lies in (1..10)"
end

if (('a'..'j') === 'c')
  puts "c lies in ('a'..'j')"
end

if (('a'..'j') === 'z')
```

```
  puts "z lies in ('a'..'j')"
end
```

This will produce following result:

```
5 lies in (1..10)
c lies in ('a'..'j')
```